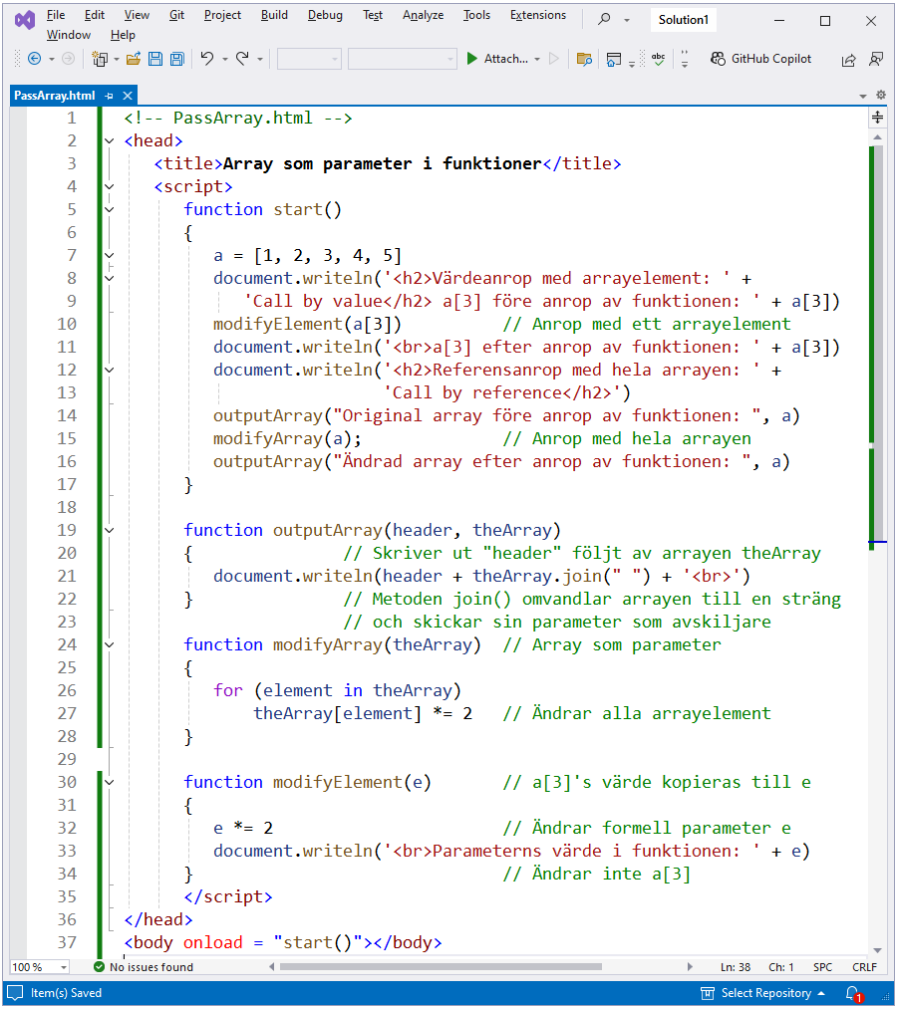
5.5 Array i funktioner



Scriptet **PassArray** skapar en array på rad **7** och initierar den samtidigt. Ett av arrayens element skickas till en funktion (rad **10**) och hela arrayen skickas till en (annan) funktion (rad **15**). Det första anropet på rad **10** är som vanligt, eftersom `a[3]` är *ett* värde. Men i det andra anropet på rad **15** är parametern en array, vilket visar att man kan skickar en array som parameter till en funktion genom att skriva arrayens namn, i det här fallet `a`, i parameterlistan. Och detta utan hakparenteser och utan uppgiften om arrayens storlek – som om arrayen vore en vanlig variabel.

Körresultatet visas på nästa sida.

Array som parameter i funktioner

I funktionens definition på raderna **24-28** tas den aktuella parametern `a` emot av den formella parametern `theArray`. Även där finns inga spår av varken hakparenteser eller av arrayens storlek. I funktionen `modifyArray()` multipliceras arrayens alla element med `2` i en `foreach`-sats (sid 98).

Som man ser i körresultatet till höger på sista raden syns denna ändring även efter anropet av funktionen i programmet. I resten av scriptet **`PassArray`** undersöks om denna ändring av arrayens element i funktionen även sysns utanför funktionen, även gäller när man gör samma sak med arrayens enskilda element `a[3]`. Körresultatet visar nämligen att detta *inte* är fallet. Vi ska nu förklara varför?

Värdeanrop: Call by value

Funktionen `modifyElement()` anropas på rad **`10`** och är definierad på raderna **`30-34`**. Vid anropet skickas elementet `a[3]` som aktuell parameter till funktionen och tas emot av den formella parameter `e`. I funktionen multipliceras `e` med `2` (rad **`32`**). Men denna ändring sysns inte *efter* anropet, när `a[3]` skrivs ut på rad **`11`**. Det är egentligen inte konstigt, eftersom `a[3]` och `e` är två olika variabler som har sin giltighet i var sitt kodområde (scope). Närmare bestämt är `e` en lokal variabel i funktionen `modifyElement()` och är inte giltig utanför funktionen. `a[3]` däremot är arrayen `a`:s fjärde element som har initierats där och gäller i hela scriptet.

Det som händer vid anropet på rad **`10`** är att `a[3]`:s *värde* som är `4`, kopieras över från minnescellen `a[3]` till minnescellen `e`. Kopians värde ändras, men detta påverkar inte originalet, vilket man ser i körresultatet. Denna metod för parameteröverföring kallas i programmering för *Värdeanrop*, på eng. *Call by value*. Beteckningen motiveras av att det är *värdet* på den aktuella parametern `a[3]` som överförs till den formella parametern `e`. Värdeanrop tillämpas automatiskt, när parametern är av enkel datatyp. En helt annan metod för parameteröverföring tillämpas när parametern är en *array*:

Referensanrop: Call by reference

Funktionen `modifyArray()` anropas på rad **`15`** och är definierad på raderna **`24-28`**. Vid anropet skickas arrayen `a` som aktuell parameter till funktionen och tas emot av den formella parameter `theArray`. I funktionen multipliceras alla element av `a` med `2` med hjälp av en `foreach`-sats (rader **`26-27`**). Sista raden i körresultatet visar att denna ändring sysns *efter* anropet, när `a` skrivs ut på rad **`16`**: Arrayens alla element är

fördubblade. Detta är märkligt, speciellt med den argumentation som framfördes vid värdeanropet och med tanke på att den aktuella och den formella parametern har olika namn: `a` och `theArray`. Förklaringen är datatypen array som inte längre är en enkel, utan en sammansatt datatyp:

Det som händer vid anropet på rad **15** är att inte arrayen `a`:s värden, utan dess *adress* skickas till funktionen. Ett annat namn för adress är *referens*. Den formella parametern `theArray` blir en referens till samma array som `a`. Minnescellerna till arrayen `a` får nu ett slags alias (referens) som pekar på samma array. När arrayens element fördubblas med hjälp av referensen `theArray`, skrivs ut dem med `a` och har med sig ändringen, vilket man ser i körresultatet. Denna metod för parameteröverföring kallas i programmering för *Referensanrop*, på eng. *Call by reference*. Beteckningen motiveras av att det är den aktuella parametern `a`:s *referens* (adress) som överförs till den formella parametern `theArray`. Referensanrop tillämpas automatiskt när parametern är en *array*, dvs en *sammansatt datatyp*.